EXPERIMENTAL EVALUATION OF APRIORI AND EQUIVALENCE CLASS CLUSTERING AND BOTTOM UP LATTICE TRAVERSAL (ECLAT) ALGORITHMS

M. Sinthuja¹, P. Aruna² and N. Puviarasan³

Department of CSE, Annamalai University, Chidambaram, India
sinthujamuthu@gmail.com, arunapuvi@yahoo.co.in, npuvi2410@yahoo.in

ABSTRACT
Frequent pattern mining is the beginning of association rule mining. Association rule mining is the strongly scrutinized techniques in data mining. The basic algorithms of Apriori and ECLAT are the most identified algorithms for mining frequent patterns in association rule mining. This paper describes the application of these two algorithms that use many to achieve maximum efficiency with regards to turnaround time and memory capacity. Both algorithms are executed using discrete data sets and are further analyzed based on their performances. The performance analysis is based on different parameters such as support, speedup etc., with different quantities of datasets.

Keywords— Apriori Algorithm; Association Rule Mining Algorithm; Data mining; ECLAT; Minimum support; Pruning.

I. INTRODUCTION
The vital logic that people were attracted by IT is the discovery of useful information from huge collection of data industry towards the domain of Data Mining [1, 2]. From the huge data, we barely explore useful knowledge for decision analysis in the business. Vast collection of data can be in distinct formats like audio, video, numbers, text, figures and hypertext formats. To perform data mining task expertise and learning are fundamental need because the victory and loss of data mining projects is highly dependent on the person who are administrating the procedure due to lack of standard protocol. The lifecycle of data mining is of six steps they are Data cleaning, Data integration, Data Selection, Data transformation, Data Mining, Knowledge discovery shown in Fig. 1.

Fig. 1. Process of KDD
Among the techniques of data mining (classification, clustering, prediction and sequential pattern discovery, etc.) The most important one is the mining association rules [3, 8]. The goal of association rule mining is to explore frequent patterns.

Two Step Approaches of Association Rule
i. Frequent Itemset Generation
ii. Rule Generation

Two basic attributes of Association Rule Mining (ARM) are
i. Support
ii. Confidence

Support(s) of an association rule is described as the fraction of records that contains the assortment of both anterior and posterior to the overall transaction in the database collection.

Confidence(c) is defined as the proportion of the number of transaction that encompasses anterior and after the overall records that contain D.

Frequent pattern mining, which is the most important field in association rule mining, was first introduced for Market Basket Analysis [4]. The goal of frequent pattern mining is to discover frequent patterns whose support is greater than or equal to the minimum support threshold. Pattern mining algorithm can be enforced on various data such as transaction databases etc. Frequent Patterns are itemsets, substructures that appear in a database with high frequency. They are Candidate generation and Pattern growth. Various types of algorithms used in association rule are shown in Fig. 2.

Fig. 2. Various types of algorithms used in association rule

Let us consider the case of the supermarket where the owner may not have input of performance of a product in the market or the most sought product. The data obtained may be used to determine the profitability of a product. In such cases, frequent pattern mining algorithms are applied.

II. LITERATURE SURVEY ON FREQUENT PATTERN MINING ALGORITHMS
Numerous techniques have been experimented for mining association rules in the research studies [13] [14, 15]. In the arena of association rule mining the Apriori algorithm is most extensively used algorithm that generates candidate patterns [6]. It is a level-wise
search. It mines frequent patterns by multiple scans of a database.

Based on Apriori algorithm, a numerous algorithm has been worked out with some improvements or adjustments such as AprioriTid algorithm [4]. It recovers more time and usage of memory is minimal. Apriori Hybrid [4], SetM (Set Oriented Mining of association rules) [15], Partition algorithm, Sampling algorithm, CARMA (Continuous Association Rule Mining algorithm) [15], DIC algorithm (prefix tree data structure) [15] are further improved Apriori algorithm which decreases the database scans.

ECLAT uses a vertical layout of a database. Each item is symbolized by a set of transaction ids called tidset [6]. It overcomes the bottleneck of apriority algorithm with regards to database scan. Rapid Association Rule mining (RARM) mentioned in [16] creates large itemsets by using a tree structure-SOTrieIT and without scanning. It does not generate candidate itemset.

Another achievement in the frequent pattern mining is FP-Growth algorithm. Han et al., introduced an energetic algorithm called FP-Growth which establishes a frequent pattern tree construction called FP-Tree It overcomes two flaws of Apriori algorithm [17]. First, it does not generate candidate patterns. Second, database scan is done only twice. It adopts divide and conquer method.

An improved frequent pattern (IFP) growth technique for discovering frequent patterns is proposed [18]. This algorithm requires lower usage of memory and it shows improved results in testing with FP-tree based algorithm.

### III. ASSOCIATION RULE MINING ALGORITHMS

Association rule mining adopts the fundamental algorithms of Apriori and ECLAT to discover effective frequent patterns. The comparison is performed for Apriori and ECLAT algorithm in terms of runtime and memory.

#### A. Apriori Algorithm

Apriori is the early generated frequent pattern mining algorithm [5, 7]. Apriori apply an iterative approach known as level-wise search [3]. It uses an effective concept called as pruning where pruning eliminates the less occurring items.

To illustrate, the transaction database is shown in Table 1, the database has ten transactions. Fig. 3 portray the generation of frequent patterns using Apriori algorithm. The first stage is analysis of database. By scanning the database, the frequency of each item is found. In the second stage, joining step is performed that is combining each item with the other items. This is called as candidate itemset 1.

**Table 1: Transaction Database**

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{a, d, e}</td>
</tr>
<tr>
<td>2</td>
<td>{b, c, d}</td>
</tr>
<tr>
<td>3</td>
<td>{a, c, e}</td>
</tr>
<tr>
<td>4</td>
<td>{a, c, d, e}</td>
</tr>
<tr>
<td>5</td>
<td>{a, e}</td>
</tr>
<tr>
<td>6</td>
<td>{a, c, d}</td>
</tr>
<tr>
<td>7</td>
<td>{b, c}</td>
</tr>
<tr>
<td>8</td>
<td>{a, c, d, e}</td>
</tr>
<tr>
<td>9</td>
<td>{b, c, e}</td>
</tr>
<tr>
<td>10</td>
<td>{a, d, e}</td>
</tr>
</tbody>
</table>

**Table 2: Candidate Itemsets and Support**

<table>
<thead>
<tr>
<th>Items</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>{a, c}</td>
<td>4</td>
</tr>
<tr>
<td>{a, d}</td>
<td>5</td>
</tr>
<tr>
<td>{a, e}</td>
<td>6</td>
</tr>
<tr>
<td>{b, c}</td>
<td>3</td>
</tr>
<tr>
<td>{c, d}</td>
<td>4</td>
</tr>
<tr>
<td>{c, e}</td>
<td>4</td>
</tr>
<tr>
<td>{d, e}</td>
<td>4</td>
</tr>
</tbody>
</table>

**Frequent patterns**

<table>
<thead>
<tr>
<th>Items</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>{a, b}</td>
<td>0</td>
</tr>
<tr>
<td>{a, c}</td>
<td>4</td>
</tr>
<tr>
<td>{a, d}</td>
<td>5</td>
</tr>
<tr>
<td>{a, e}</td>
<td>6</td>
</tr>
<tr>
<td>{b, c}</td>
<td>3</td>
</tr>
<tr>
<td>{b, d}</td>
<td>1</td>
</tr>
<tr>
<td>{b, e}</td>
<td>1</td>
</tr>
<tr>
<td>{c, d}</td>
<td>4</td>
</tr>
<tr>
<td>{c, e}</td>
<td>4</td>
</tr>
</tbody>
</table>

Combining the item ‘a’ with ‘b’, then ‘a’ with ‘c’. Similarly, combining all the items. Again, scanning the database to find the occurrence of combinations of each item. The count of {a, b} is 0 and {a, c} is 4. Likewise find the support for each combination. In the succeeding stages concept of pruning is performed. Here the minimum support is greater than 2. The itemset greater than 2 is taken for the next step and <= 2 are eliminated. So itemsets {a, c} {a, d}{a, e}{b, c}{c, d}{c, e}{d, e} are taken for the next step called as frequent itemset 1. In the next stage perform join step to find candidate itemset 2. Again the database is scanned to find the frequency for each itemset. Apply pruning and eliminate the itemset less than 2 and equal to 2. Finally the frequent itemsets are {a,c,d: 3}{a,c,e: 3}{a,d,e: 4}.

The drawback of Apriori algorithm is that if the frequent patterns are longer then, the algorithm must perform more iteration. Thus, the performance of

Minimum Support >2
Apriori reduces. Multiple scanning of database is needed. It is not useful for real time applications. The duration of the protocol is long.

**Algorithm 1: Apriori**

| Input: Database of Transactions D= \{t_1, t_2 \ldots t_n\}  
| Set if Items I= \{I_1, I_2, \ldots I_k\} 
| Frequent (Large) Itemset L  
| Support 
| Confidence  
| Output: Association Rule satisfying Support & Confidence  
| Method:  
| Step 1: C_1 = Itemsets of size one in I; 
| Step 2: Determine all large itemsets of size L_1, L_1;  
| Step 3: i = 1;  
| Step 4: Repeat  
| Step 5: i = i + 1;  
| Step 6: C_i = Apriori-Gen(L_{i-1});  
| Step 7: Apriori-Gen(L_{i-1})  
| Step 8: Generate candidates of size i+1 from large itemsets of size i.  
| Step 9: Join large itemsets of size i if they agree on i-1.  
| Step 10: Prune candidates who have subsets that are not large.  
| Step 11: Count C_i to determine L_i;  

### B. Equivalence Class Clustering and Bottom Up Lattice Traversal (ECLAT)

Improvised algorithm of Apriori is Equivalence Class Clustering and Bottom up Lattice Traversal (ECLAT) [6,7,8]. ECLAT proposed in [11,12]. ECLAT overthrown the limitation of Apriori in case of database scan, it requires only one database scan. This algorithm converts horizontal database to vertical layout where the Apriori algorithm uses horizontal database [9,10].  

Like Apriori algorithm it has the concept of candidate generation and pruning.

Fig. 4 illustrates the construction of ECLAT with sample database. Table 2 represents the sample database which is horizontal in structure. The first challenge is to convert the horizontal database into a vertical one which is shown in Figure 3. Each item is simulated by group of transaction ids which is called tidset and finding the occurrence of each item.  

The number of items in the dataset is scrutinized. The frequencies of each item with respect to tids are counted. Each item is combined with another item known as join step. Let us consider items ‘a’ and ‘b’. The frequency of combination ‘a’ and ‘b’ is found out in the vertical layout. There is no occurrence of combination ‘a’ and ‘b’ so its frequency is \{a, b: 0\}.

The drawback of ECLAT is the requirement of virtual memory to process the transaction.

**Algorithm 2: ECLAT**

| Input: F = \{I_1, I_2 \ldots I_n\} frequent k Itemsets  
| Terminology:  
| (i) \(F_k\) is defined as database having \(F_k = \{I_1, I_2, \ldots, I_n\}\)  
| (ii) \(\Phi\) denotes the itemsets .where itemsets means collection of items in database \(F_k\)  
| (iii) \(I_1\) and \(I_j\) both should be from same equivalence class  
| Output: \(F_{k+1}\) Frequent Item Sets  
| Bottom-Up (\(F_k\)):  
| Step 1: for all \(I_i \in F_k\) do  
| Step 2: \(F_{k+1} = \Phi\);  
| Step 3: for all \(I_i \in F_k\), \(i < j\) do  
| Step 4: \(N = I_i \cap I_j\); // Both should be from same equivalence class  
| Step 5: if \(N.sup >= minsup\) then  
| Step 6: \(F_{k+1}^= = F_{k+1}^= \cup \{N\}\); \(F_{k+1}^\Phi = F_{k+1}^\Phi \cup \{N\}\)  
| Step 7: end;  
| Step 8: if \(F_{k+1}^\Phi = \Phi\) then  
| Step 9: Bottom-Up (\(F_{k+1}\));  
| Step 10: end;  

Table 2: Transaction DB

**Vertical layout of the initial database**

Each item is symbolized by set of transaction ids called tidset
### IV. PERFORMANCE EVALUATION

#### A. Dataset Description

The above specified algorithms are implemented using different standard datasets of different domains namely mushroom, supermarket, German, Eucalyptus, Primary tumor where it is available in Tunedit Machine Learning Repository. Mushroom accommodates 8124 instances and 23 attributes i.e. cap shape, cap surface, cap color, class etc. Supermarket contains 4627 transactions and 217 attributes especially Grocery, baby needs, coupons, breakfast food etc. German consists of 1000 instances and 21 attributes especially personal status and sex, telephone, housing, property etc. Primary tumor accommodates 339 transactions and 18 attributes i.e. brain, skin, neck, abdominal, liver, age, sex etc. Eucalyptus contains 736 transactions and 20 attributes namely locality, latitude, altitude and year etc.

#### B. Experimental Results

In this research paper, we compare the performances of ECLAT and Apriori algorithms. Different support levels are used for each datasets of algorithm. The experiments are conducted on Intel® corei3™ CPU, 2.13 GHz, and 2GB of RAM computer.

Fig. 5 illustrates the graphical user interface of Apriori which displays the details of minimum support, dataset and algorithm used. It displays the amount of time required to execute the algorithm, utilization of memory space and the number of frequent patterns.

The frequent patterns generated by Apriori algorithm using mushroom dataset is shown in Fig. 6. Frequent patterns explored from the experiment are based on support.

---

Fig. 5. Graphical user interface

Fig. 4. Steps to derive frequent patterns using ECLAT algorithm

Fig. 6. Frequent Patterns found using Apriori for mushroom dataset
Graphical user interface of ECLAT is displayed in Fig. 7 which contains the information of user specified minimum support, type of dataset and algorithm used. It shows the runtime taken to execute the algorithm, usage of memory space and the total number of frequent patterns.

Regarding Fig. 8, the frequent pattern generated by ECLAT algorithm using mushroom dataset is displayed. Frequent patterns explored from the experiment are based on support.

Table 3 depicts the runtime of each algorithm with different datasets. The processing time of mushroom dataset for Apriori is 211ms and for ECLAT is 265ms. The processing time of supermarket dataset for Apriori is 30ms and ECLAT is 35ms. The processing time of german dataset for Apriori is 1211ms and ECLAT is 303ms. As the minimum support increases ECLAT is poorer in runtime when compared to Apriori. While minimum support decreases Apriori perform faster than ECLAT in terms of execution time.

Table 4 depicts the memory usage of both algorithms with respect to different datasets. Memory space required for Apriori is 12.40mb and ECLAT is 21.77 mb for mushroom dataset. Usage of memory for Apriori is 8.90mb and ECLAT is 17.01mb for supermarket dataset.

Table 3: Execution time of instances

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Support</th>
<th>Apriori</th>
<th>ECLAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mushroom</td>
<td>0.4</td>
<td>211</td>
<td>265</td>
</tr>
<tr>
<td>Supermarket</td>
<td>0.4</td>
<td>30</td>
<td>35</td>
</tr>
<tr>
<td>German</td>
<td>0.1</td>
<td>1211</td>
<td>303</td>
</tr>
<tr>
<td>Eucalyptus</td>
<td>0.1</td>
<td>102</td>
<td>32</td>
</tr>
<tr>
<td>Primary</td>
<td>0.3</td>
<td>334</td>
<td>162</td>
</tr>
<tr>
<td>tumor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>1888</td>
<td>797</td>
</tr>
</tbody>
</table>

Fig. 9 illustrates the performance comparison of Apriori and ECLAT algorithms with different support threshold value for different datasets. Runtime of Apriori is faster than ECLAT in the case of higher minimum support threshold value. While in the case of lower minimum support Apriori is poor runtime when compared to ECLAT.

Table 4: Memory usage

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Memory Usage in mb</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apriori</td>
<td>ECLAT</td>
</tr>
<tr>
<td>Mushroom</td>
<td>12.40</td>
</tr>
<tr>
<td>Supermarket</td>
<td>8.90</td>
</tr>
<tr>
<td>German</td>
<td>25.43</td>
</tr>
<tr>
<td>Eucalyptus</td>
<td>93.03</td>
</tr>
<tr>
<td>Primary tumor</td>
<td>105.51</td>
</tr>
<tr>
<td>Average</td>
<td>255.59</td>
</tr>
</tbody>
</table>

Fig. 10 portrays the comparative results of Apriori and ECLAT in case of memory usage. Apriori requires less memory than ECLAT in case of maximum minimum support. While using lower minimum support Apriori requires more memory space than ECLAT.
IV. CONCLUSION

In this paper the researcher surveyed frequent pattern mining algorithms namely Apriori and Equivalence Class Clustering and Bottom Up Lattice Traversal (ECLAT) with distinct database. The goal of Apriori and Equivalence Class Clustering and Bottom up Lattice Traversal is to explore frequent patterns. It is found that Apriori used join and prune technique and ECLAT works on vertical datasets. The major drawback of Apriori is generating large number of candidate patterns and huge number of database scan. The major limitation of ECLAT requires virtual memory to process the transaction.

The average runtime of Apriori and ECLAT shows that Apriori performs worst in runtime when compared to ECLAT in case of different minimum support. The memory usage of both algorithms results that ECLAT is poorer than Apriori.

REFERENCES